On-line computation and maximum-weighted hereditary subgraph problems

Marc Demange, Bernard Kouakou, Eric Soutif

Centre d'Economie de la Sorbonne
UMR 8174

Cahiers de la MSE

On-line computation and maximum-weighted hereditary subgraph problems

Marc DEMANGE

Bernard KOUAKOU

Eric SOUTIF

2006.34

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Maison des Sciences Économiques, 106-112 boulevard de L'Hôpital, 75647 Paris Cedex 13
http://mse.univ-paris1.fr/Publicat.htm

ISSN : 1624-0340

On-line computation and maximum-weighted hereditary subgraph problems

Marc Demange[1], Bernard Kouakou[2] and Éric Soutif[3]

[1] ESSEC, SID department, Avenue Bernard HIRSH, BP 105, F-95021 Cergy Pontoise Cedex, France, demange@essec.fr,
[2] CERMSEM, Université Paris 1, 106–112 bd de l'Hôpital, F-75013 Paris, France, kouakou@univ-paris1.fr
[3] CEDRIC, CNAM, 292 rue Saint-Martin, F-75003 Paris, France, soutif@cnam.fr

Abstract. In this paper, we study the on-line version of maximum-weighted hereditary subgraph problems. In our on-line model, the final instance-graph (which has n vertices) is revealed in t clusters, $2 \leq t \leq n$. We first focus on the on-line version of the following problem: finding a maximum-weighted subgraph satisfying some hereditary property. Then, we deal with the particular case of the independent set. For all these problems, we are interested in two types of results: the competitivity ratio guaranteed by the on-line algorithm and hardness results that account for the difficulty of the problems and for the quality of algorithms developed to solve them.

keywords: on-line algorithm, hereditary property, independent set, competitivity ratio.

1 Introduction

On-line computation aims to solve combinatorial problems with the constraint that the instance is not a priori completely known before one begins to solve it. In other words, the data-set is revealed step-by-step and one has, at the end of each step, to irrevocably decide on the final solution dealing with this step. On-line algorithms deal with a large class of problems subjected to time constraints (one must take decisions before knowing all data). An on-line problem is defined by:

- a combinatorial optimisation problem,
- a set of rules R describing how the final instance will be revealed,
- a set of rules R' defining what kind of decisions are allowed.

1.1 Maximum weighted hereditary subgraph problems

In this paper, we deal with on-line versions of the problem of finding, in a weighted graph G, a maximum weighted subgraph satisfying a non-trivial hereditary property π.

Definition 1. *Let us consider a graph-property π from the set of graphs to $\{0,1\}$. π is said to be satisfied for a graph G if and only if $\pi(G) = 1$. Property π is hereditary if, whenever it is satisfied by a graph, it is also satisfied by each of its induced subgraph; it is non-trivial if it is true for an infinite number of graphs and false for an infinite number of graphs.*
If $G = (V, E)$ is a graph, then a set of vertices $V' \subset V$ is said to satisfy π if and only if the induced subgraph $G[V']$ satisfies π.

Since π is assumed to be non trivial, then for every n there exists a graph of order at least n satisfying π; which implies that an isolate vertex satisfies π.

Definition 2. *Given a graph $G = (V, E)$, the maximum hereditary subgraph problem, HG, consists of finding a subgraph of maximum order of G satisfying a given non-trivial hereditary property π.*

WHG denotes the weighted version of HG: the input is a weighted graph (each vertex x has a weight $w(x)$) and one looks for a maximum-weighted subgraph satisfying π. Some on-line versions of HG have been studied in [3]. This paper is devoted to extend the results to the weighted case.

1.2 On-line models and competitive analysis

Let $LWHG$ denote the on-line version of WHG. By using the framework described in [7], it is defined by (WHG, R, R'). Given a problem, we can pass from an on-line version to another by changing only the rules R and R'. For $LWHG$ we consider the same model as in [3]:
R: the graph G is revealed in t steps. At each step, a weighted subgraph $G_i = (V_i, E_i)$ (called cluster) of G is revealed together with the edges linking the vertices of G_i and already revealed vertices. Consequently at step i, the graph already revealed is $G[V_1, \cdots, V_i]$, the subgraph of the whole instance induced by $V1, \cdots, V_i$.
R': at each step, one irrevocably decides which new vertices are introduced in the solution.
The quality of an on-line algorithm is expressed by the competitive ratio defined below.

Let us consider a maximization on-line problem P, an instance σ of P and an algorithm **LA** for solving P. Furthermore, we consider a function $c_{\mathbf{LA}}(n)$ (n denotes the order of the instance). Let $\mathbf{LA}(\sigma)$ denote the value of the solution of P computed by the algorithm LA and $\beta(\sigma)$ the value of the solution returned by an optimal algorithm knowing beforehand the final instance. The algorithm **LA** is said to guarantee the competitivity ratio $c_{\mathbf{LA}}(n)$ if, for any instance σ of P:
$c_{\mathbf{LA}}(n) \times \beta(\sigma) \leq \mathbf{LA}(\sigma)$ ($c_{\mathbf{LA}}(n) \times \mathbf{LA}(\sigma) \leq \beta(\sigma)$ for a minimization problem).
LA is also said to be $c_{\mathbf{LA}}(n)$-competitive.
In what follows, we supposed that the number of vertices and the total weight of the final graph are known at the beginning of the on-line process. In the opposite

case, it is easy to see that no interesting result can be found; only the trivial ratio $\frac{W_{min}}{W(G)-W_{min}}$ can be guaranteed (where $W(G)$ and W_{min} denote respectively the total and the minimum weight of the final graph G).

1.3 Approximation ratio

In the case of approximation, given a polynomial-time algorithm **A** computing, for every instance σ of an $NP-hard$ (off-line) minimization problem (the maximization case is defined in a similar manner), a feasible solution, **A** is said to guarantee an approximation ratio of ρ_A if, for every instance σ of order n, the approximation value $\mathbf{A}(\sigma)$ satisfies: $\rho_A(n) \times \beta(\sigma) \leq \mathbf{A}(\sigma)$ where $\beta(\sigma)$ denotes the value of the optimal solution of the instance.
For $LWHG(t)$ (t *steps with* $t << n$), we are interested in links between off-line and on-line algorithms. A central question in this work is how it is possible to exploit (polynomial-time) algorithms in order to devise (polynomial-time) on-line algorithm and moreover, is it possible to transfer performance guarantees from off-line to on-line framework?
For this study, we suppose we are given a $\rho(n)$-approximation algorithm FWA for WHG such that the following hypotheses **H** hold:

1. The considered approximation algorithm **FWA** solving WHG is supposed to guarantee ratio of the form $\frac{f(n)}{n}$, where f is an increasing function in n beyond k.
2. Algorithm **FWA** satisfies $w(\mathbf{FWA}(G)) \geq \frac{w(V)}{n}$, $\forall G$ (the ratio $1/n$ is always guaranteed, i.e $f(n) \geq 1$);

These hypotheses are not restrictive (see also [3]); in particular item 1 is satisfied for every known approximation ratios for HG and WHG: both problems can be polynomially approximated within $O(\log n/n)$ and this ratio can be improved to $O(\log^2 n/n)$ for maximum independent set [4, 2]. Condition 2 is also natural since the ratio $\frac{w(V)}{n}$ can always be guaranteed.

In section 2, we show that $LWHG(t)$ reduces to WHG. More precisely, we show that a (polynomial-time) $f(n)/n$-approximation algorithm leads to a (polynomial) on-line algorithm with competitive ratio $c_{\mathbf{LWA}}(G) \geq \frac{1-f(n)^{1/t}}{1-f(n)} \frac{f(n)}{n}$. For $t = 2$, it leads to a result of [3] which was obtained by adapting to $LWHG(2)$ the methodology used for $LHG(t)$. The proof of our result is totally different while the on-line algorithm is quite similar. At each step, it computes a solution of the problem restricted to the new cluster and decides either to include the whole solution performed if its value is sufficiently good or to reject it. We call such an algorithm a *threshold algorithm*.
In section 3, we perform lower and upper bounds for a class of algorithms including threshold ones. The main result is shown in the case where π is either *clique* or *independent set*. Finally, section 4 deals with on-line independent set problem for which we propose an hardness result that bounds below the competitive ratio of any algorithm (not only threshold ones). It points out that, for this problem, threshold algorithms are almost optimal.

1.4 Notations

We will consider only simple graphs $G = (V, E)$, $n = |V|$ denotes the order of G. Every edge in G will be denoted either by $(i, j) = (j, i)$ or simply by $ij = ji$. $\bar{G} = (V, \bar{E})$, $\bar{E} = \{(i, j), i \neq j, ij \notin E\}$ denotes the complement of G. If $V' \subset V$, $G[V']$ is the subgraph of G induced by V'. An independent set of G is a set of vertices which are mutually not linked by an edge: $V' \subset V$, $\forall (i, j) \in V' \times V', ij \notin E$, in other words, $G[V']$ has no edge. A clique of G is a set of vertices such that $G[V']$ is a complete graph or, equivalently, V' is an independent set of $\bar{G}$. $w(G)$ denotes the sum of the weights of the vertices of G. The on-line version of any hereditary weighted graph problem in which the final instance is revealed in t steps will be denoted by $LWHG(t)$. $LWIS$ denotes the on-line version of the weighted independent set problem.

2 Competitive ratio for $LWHG$

Recall that the total number of vertices and the total weight are supposed to be known in advance. This section is devoted to prove the following result:

Theorem 1. *Suppose that WHG admits a polynomial-time $\rho(n)-$approximation algorithm **FWA** with $\rho(n) = \frac{f(n)}{n}$. Then, under the hypotheses H, there exists an on-line polynomial-time algorithm **LWA**, for $LWHG$, achieving for all graph G of order n a competitive ratio of:*

$$c_{\boldsymbol{LWA}}(G) \geq \frac{1 - f(n)^{1/t}}{1 - f(n)} \rho(n).$$

Moreover, for $\epsilon > 0$ and n large enough: $c_{\boldsymbol{LWA}}(G) \geq (1 - \epsilon) \frac{f(n)^{1/t}}{n}$.

Proof. Let us consider **LWA**, the following algorithm which receives the input-graph G in t subgraphs $G_1, G_2, \ldots, G_t$ of respective order $n_1, n_2, \ldots, n_t$, and returns the solution $\mathbf{LWA}(G)$ for the $LWHG$ problem :

Algorithm 1

1. $i \leftarrow 1$;
2. $W \leftarrow w(G)$;
3. $r \leftarrow n$;
4. while $w(\mathbf{FWA}(G_i)) < \frac{W - w(G_i)}{r - n_i} f(r - n_i)^{1/t}$ and $i < t$ do
5. $\quad W \leftarrow W - w(G_i)$;
6. $\quad r \leftarrow r - n_i$;
7. $\quad i \leftarrow i + 1$;
8. end while
9. return $\mathbf{LWA}(G) \leftarrow \mathbf{FWA}(G_i)$;

Remark 1. This is a "threshold-algorithm" using the threshold $\frac{W-w(G_i)}{r-n_i} f(r-n_i)^{1/t}$. It is polynomial since algorithm **FWA** is polynomial-time.

For $i \leq t-1$, we denote $R_i = G[V_{i+1} \cup \cdots \cup V_t]$, and $r_i = n_{i+1} + \cdots + n_t$. Let us denote by $k \leq t$ the value of i at the end of the while loop. We will consider two cases whether $k < t$ or $k = t$. In the first case, the following statements hold:

$$w(\mathbf{FWA}(G_i)) < \frac{w(R_i)}{r_i} f(r_i)^{1/t} \qquad \forall i < k \tag{1}$$

$$w(\mathbf{FWA}(G_k)) \geq \frac{w(R_k)}{r_k} f(r_k)^{1/t} \geq \frac{w(R_k)}{r_k} \tag{2}$$

In the second case, only relation 1 holds.
We first point out the following result:

Lemma 1.

$$\forall i < k,\ w(\boldsymbol{FWA}(G_i)) \leq f(n)^{\frac{k-i}{t}} w(\boldsymbol{FWA}(G_k))$$

Proof. (of lemma 1)
Let us first consider that $k < t$. For $i < k$, since $w(R_i) = w(G_{i+1}) + \cdots + w(G_k) + w(R_k)$, relation 1 becomes

$$w(\mathbf{FWA}(G_i)) < \frac{w(G_{i+1}) + \cdots + w(G_k) + w(R_k)}{n_{i+1} + \cdots + n_k + r_k} f(r_i)^{1/t}.$$

By using item 1 of **H**, we deduce:
$w(\mathbf{FWA}(G_i)) \leq \frac{n_{i+1} w(\mathbf{FWA}(G_{i+1})) + \cdots + n_k w(\mathbf{FWA}(G_k)) + r_k w(\mathbf{FWA}(G_k))}{n_{i+1} + \cdots + n_k + r_k} f(n)^{1/t}$
which implies $w(\mathbf{FWA}(G_i)) \leq f(n)^{1/t} \sup_{i+1 \leq q \leq k} w(\mathbf{FWA}(G_q))$.

Then, a simple backward induction concludes the result. The case $k = t$ is simmilarly solved by putting $w(R_t) = 0$ and $r_t = 0$, which concludes the proof of lemma 1.

Let us continue the demonstration of the theorem:
Case 1: algorithm stops with $k < t$. Heredity implies that $\beta(G) \leq \beta(G_1) + \cdots + \beta(G_k) + \beta(R_k)$, moreover, $\beta(R_k) \leq\ w(R_k) \leq\ \frac{r_k}{f(r_k)^{1/t}} w(\mathbf{FWA}(G_k))$, which implies:
$\beta(G) \leq\ \rho(n_1)^{-1} w(\mathbf{FWA}(G_1)) + \cdots + \rho(n_k)^{-1} w(\mathbf{FWA}(G_k)) + \frac{r_k}{f(r_k)^{1/t}} w(\mathbf{FWA}(G_k))$
By using the increasing of f and the decreasing of $\rho(n) = f(n)/n$ (item 1 of hypothesis H), we deduce:
$\beta(G) \leq \rho(n)^{-1} \big(w(\mathbf{FWA}(G_1)) + \cdots + w(\mathbf{FWA}(G_k)) + f(n)^{\frac{t-1}{t}} w(\mathbf{FWA}(G_k)) \big)$
Then lemma 1 implies

$$\beta(G) \leq \rho(n)^{-1} \big(f(n)^{\frac{t-2}{t}} + \cdots + f(n)^{\frac{1}{t}} + 1 + f(n)^{\frac{t-1}{t}}) w(\mathbf{FWA}(G_k)) \big)$$

and finally

$$\beta(G) \leq \rho(n)^{-1} \frac{1 - f(n)}{1 - f(n)^{\frac{1}{t}}} w(\mathbf{FWA}(G_k))$$

The related ratio is:

$$\frac{w(S)}{\beta(G)} \geq \rho(n)\frac{1-f(n)^{\frac{1}{t}}}{1-f(n)}.$$

Case 2: the algorithm runs until the t^{th} iteration.
By using similar arguments to the first case, we successively get:
$\beta(G) \leq \beta(G_1) + \cdots + \beta(G_t)$
$\beta(G) \leq \rho(n_1)^{-1} w(\mathbf{FWA}(G_1)) + \cdots + \rho(n_t)^{-1} w(\mathbf{FWA}(G_t))$
$\beta(G) \leq \rho(n)^{-1}\big((w(\mathbf{FWA}(G_1)) + \cdots + w(\mathbf{FWA}(G_t))\big)$
$\beta(G) \leq \rho(n)^{-1}\big(f(n)^{\frac{t-1}{t}} + \cdots + f(n)^{\frac{1}{t}} + 1\big) w(\mathbf{FWA}(G_t))$ so $\frac{w(S)}{\beta(G)} \geq \rho(n)\frac{1-f(n)^{\frac{1}{t}}}{1-f(n)}$
Since f is supposed to infinitly increase, the asymptotic equivalent immediately follows.

We deduce from approximation results for WHG and WIS:

Corollary 1. *For fixed values of t,*

i *$LWHG(t)$ admits a polynomial $O((\log n)^{1/t}/n)$-competitive algorithm;*
ii *$LWIS(t)$ admits a polynomial $O((\log n)^{2/t}/n)$-competitive algorithm.*

If we consider not only polynomial-time algorithms, the result also holds with $\rho(n) = 1$:

Corollary 2. *For fixed values of t, $LWHG(t)$ (and also $LWIS(t)$) admits a $O(n^{1/t-1})$-competitive algorithm.*

3 Limit of threshold-algorithms (hardness result)

In this section, we first suppose that π is either a *clique* or an *independent set*.

Let us consider an approximation algorithm **FWA** for WHG satisfying the set of hypotheses **H**. In [1], for problem $LWHG(2)$, a lower bound of $2\sqrt{1+\mu}\frac{\sqrt{f(n)}}{n}$ for the competitive ratio of threshold-algorithm is devised by assuming the following hypothesis $H'(\mu)$:

$H'(\mu)$**:** There exists G_1, a graph of order n_1 such that $\beta(G_1)\rho(n_1) \leq \mathbf{FWA}(G_1) < (1+\mu)\beta(G_1)\rho(n_1)$.

If μ is close to 0, $H'(\mu)$ means that the approximation ratio of the algorithm **FWA** cannot be significantly improved.

We show that this result can be generalized for any $t \geq 2$; moreover, it shows that the analysis performed in the proof of Theorem 1 is asymptocally tight.

Proposition 1. *If π is either clique or independent set and if **FWA** satisfies $H'(\mu)$ for a given approximation ratio $\rho(n) = f(n)/n$ and $\mu > 0$, then the corresponding threshold-algorithm (algorithm 1, section 2) cannot guarantee the competitive ratio $t(1+\mu)^{1-\frac{1}{t}}\frac{f(n)^{\frac{1}{t}}}{n}$.*
The result holds even if the sequence of the weights is known at the beginning of the on-line process and if clusters are of the same size.

Proof. FWA satisfies $H'(\mu)$. So there exists a graph G_1 of order n_1 such that $\beta(G_1)\rho(n_1) \leq \mathbf{FWA}(G_1) < (1+\mu)\beta(G_1)\rho(n_1)$. We consider $t-1$ real numbers $x_1, x_2, \ldots, x_t$ strictly positive such that:

$$\mathbf{FWA}(G_1) < x_t < \cdots < x_2 < x_1 < (1+\mu)\beta(G_1)\rho(n_1).$$

Set $w_k = \frac{nx_k}{t}(1+\mu)^{\frac{k-t-1}{t}} f(n)^{\frac{1-k}{t}}, \forall k \geq 2$.

We apply the algorithm **LWA** to a graph of $n = tn_1$ vertices and of total weight $W = w(G_1) + w_2 + \cdots + w_t$, and we suppose that G_1 is revealed at the first step. Then we apply the following strategy to reveal a final graph having n vertices and whose total weight is W so that the algorithm cannot return a good solution (actually the worst one).

1. For $i < t$, if the algorithm selects some vertices in G_i, we reveal, for $k = 1 \cdots t-i$, clusters G_{i+k} so that : for the *independent set* problem, G_{i+k} is an independent set of order n_1; every vertex of G_{i+k} is of weight $\frac{w_{i+k}}{n_1}$ and is linked to all vertices in $G[V_1 \cdots V_i]$, i.e vertices revealed at steps $1, \cdots, i$. (For the *clique* problem, G_{i+k} will be a clique with no link with $G[V_1 \cdots V_i]$).

2. For $i < t$, if the algorithm does not select vertices in G_i, we reveal G_{i+1} a graph of order n_1 so that: for the independent set problem, G_{i+1} is a clique with vertices of weight $\frac{w_{i+1}}{n_1}$ and is linked to all vertices in $G[V_1 \cdots V_i]$. (With a similar construction one can deal with the case of the *clique* problem).

We then distinguish the following three cases (recall that vertices of the final solution returned by algorithm LWA are all in a same cluster G_i):
i)We first consider the case where the solution returned by **LWA** contains vertices of G_1.
ii) Then, we deal with the case where the first selected vertices belong to G_i, with $1 < i < t$. In this case, the result returned by **LWA** is $\mathbf{FWA}(G_i)$ since rule 1 is applied to i.
iii) Finally, we study the case where the algorithm does not select vertices in the subgraphs $G_1, G_2, \ldots, G_{t-1}$ (so the algorithm returns $\mathbf{FWA}(G_t)$). Rule 2 is applied at step $t-1$.
In three cases one can establish that $\frac{\mathbf{LWA}(G)}{\beta(G)} < t(1+\mu)^{1-\frac{1}{t}} \frac{f(n)^{\frac{1}{t}}}{n}$.
We have thus shown that there exists a graph having n vertices such that $c_{\mathbf{LWA}}(n) < t(1+\mu)^{1-\frac{1}{t}} \frac{f(n)^{\frac{1}{t}}}{n}$.

Corollary 3. *Under hypotheses H and $H'(\mu)$, for any ϵ and for n large enough, we have:*

$$(1-\epsilon)\frac{f(n)^{\frac{1}{t}}}{n} < c_{\boldsymbol{LWA}}(n) < t(1+\mu)\frac{f(n)^{\frac{1}{t}}}{n}$$

This bound is asymptotically tight since the ratio between the upper and the lower bounds is $t\frac{(1+\mu)}{1-\epsilon}$.

4 On-line maximum-weighted independent set problem ($LWIS$)

Theorem 1 holds for maximum independent set problem. The lower bound devised for threshold algorithms also applies to this problem. Roughly speaking, those results show that the only way to improve the competitive ratio of algorithm 1 (LWA) for $LWIS$ is to improve the performance guarantee of the off-line algorithm **FWA** used by **LWA**. Netherless, this method is limited by the bound $O(n^{1/t-i})$ obtained by using an optimal algorithm as **LWA**. This raises the following question: is it possible to get a best competitive ratio by using another type of algorithms? In this section, we bring a negative answer to this question.

Theorem 2. *Let **LWA** be an on-line algorithm solving $LWIS$ for $t \geq 2$ (the graph is revealed in t clusters). Assume that the weights of clusters are known by the algorithm as soon as the game starts, then its competitivity ratio $c_{\boldsymbol{LWA}}$ satisfies for every n: $c_{\boldsymbol{LWA}}(n) \leq t\frac{n^{\frac{1}{t}}}{n}$.*

Proof. Let **LWA** be an on-line algorithm solving $LWIS$; consider $t \geq 2$ and $n = kt$, $k \geq 2$. Set $w_i = k^{1-\frac{i-1}{t}}$; the total weight of the final graph is then $W = \frac{k-1}{k^{\frac{1}{t}}-1} + q$ (if we suppose $k > 1$, i.e. $n > 2t-2$).

We apply algorithm **LWA** to a graph of order n and of total weight W. A first cluster consisting of a clique of $q+1$ vertices of weight 1, is revealed. Then, we apply the following strategy.

1. If at step $i < t$, **LWA** has not selected any vertex yet; a clique G_{i+1} of k vertices is revealed. Each vertex of the clique is of weight $\frac{1}{k^{\frac{i}{t}}}$ and is not linked to vertices already revealed.
2. If **LWA** selects some vertices at the step i, then clusters $G_{i+1}, G_{i+2}, \ldots, G_t$ are independent sets of size k. Their vertices are respectively of weights $\frac{1}{k^{\frac{i}{t}}}, \ldots, \frac{1}{k^{\frac{t-1}{t}}}$, and are (all) linked to an already selected vertex.

We distinguish two cases.

<u>*Case 1*</u>: rule 2 has never been used, namely the algorithm has only taken some vertices in the last cluster which is a clique; so $w(\mathbf{LWA}(G)) = \frac{1}{k^{\frac{t-1}{t}}}$.

Now $\alpha(G) \geq \alpha(G_1) = 1$, so $\frac{w(\mathbf{LWA}(G))}{\alpha(G)} \leq k^{\frac{1-t}{t}}$.

<u>*Case 2*</u>: rule 2 has been used at the step i, $i < t$. It is clear that only one vertex of weight $\frac{1}{k^{\frac{i-1}{t}}}$ has been selected (in G_i) since only cliques have been revealed until the i^{th} step and all the next vertices are linked to an already selected vertex: $w(\mathbf{LWA}(G)) = \frac{1}{k^{\frac{i-1}{t}}}$.

Moreover, $\alpha(G) \geq \alpha(G_{i+1}) = \frac{k^{\frac{i}{t}}}{k}$, So $\frac{w(\mathbf{LWA}(G))}{\alpha(G)} \leq k^{\frac{1-t}{t}}$.

In conclusion, we get (since $n = kt$): $c_{\mathbf{LWA}}(n) \leq k^{\frac{1-t}{t}} \leq (\frac{n}{t})^{\frac{1-t}{t}}$

So, $c_{\mathbf{LWA}}(n) \leq t\frac{n^{\frac{1}{t}}}{n}$.

5 Concluding remarks

This work points out that results for LGH and $LWHG(2)$ can be generalized to the weighted class $LWHG$. In the off-line case, HG and WHG are equivalently solved by polynomial-time algorithms ([2, 7]). The situation is rather different in on-line framework.
Indeed, the comparison between theorem 1 and result of [3] brings to the fore a gap between competitive ratio of $LHG(t)$ and $LWHG(t)$, which is asymptotically: $\frac{\rho_{LWIS}}{\rho_{LIS}} \sim f(n)^{\frac{1}{t}-\frac{1}{2}}\sqrt{t}$. More generally, weights induce many questions in on-line framework. In this work, we focused on an on-line model in which weights are revealed on-line together with vertices. But one can also study either a model where the sequence of the weights is revealed at the beginning while the graph is revealed on-line or a model where the graph is known in advance and weight are on-line.
Such models being particular cases of the ones we deal with in this paper, the competitive analysis performed in Theorem 1 remains valid in both cases. Let us now consider this question from the hardness results point of view. By revisiting the proof of theorem 2, let us point out of that weights are known beforehand. So this hardness result also holds for a model in which weights are removed from the on-line process.
Let us now consider a dual situation where the graph is fixed and weights are given on-line. The following proposition shows that the same hardness result raises for this model:

Proposition 2. *Let **LWA** be an on-line algorithm solving the problem $LWIS$ for $t \geq 2$ (the set of weights is completely revealed in t steps); the total weight of each cluster is known at the beginning of the process. Then, there exists a graph of order n for which the competitivity ratio $c_{\mathbf{LWA}}$ satisfies: $c_{\mathbf{LWA}}(n) \leq t\frac{n^{\frac{1}{t}}}{n}$.*

Proof. We set $n = tk$ where $k \geq 2,\ k \in N$. $w_i = k^{1-\frac{i-1}{t}}$, $n_i = k$, $\forall i \geq 1$. So the total weight of the final graph is $W = k\frac{k-1}{k^{\frac{1}{t}}-1}$.
This time, we apply the algorithm **LWA** to a complete graph of order n and of weight W. The first cluster contains $(q+1)$ weights, each of them being 1. Then we use the following strategy.

1. If at step $i < t$, **LWA** has not selected any vertex yet, we reveal a set of k identical weights $\frac{w_{i+1}}{k} = \frac{1}{k^{\frac{i}{t}}}$, in order to form the cluster G_{i+1} of total weight $w_{i+1} = k^{1-\frac{i}{t}}$.

2. If **LWA** selects vertices at step i, then the next clusters, $G_{i+1}, G_{i+2}, \ldots, G_t$ (each of them has k vertices) are such that:
for each of the clusters, $k-1$ vertices have a null weight and only one vertex supports the weight of the whole cluster.

We conclude by using the same arguments as in Theorem 2.

Let us finally underline a main difference between Theorems 1 and 2. The first one is only valid for a class of algorithms, nevertheless, it gives some informations not only about general algorithms but also about polynomial-time ones while the second one does not allow us to take complexity considerations into account. Theorem 1 brings to the fore that a threshold algorithm parametrized by an exact off-line algorithm is almost optimal among on-line algorithms. An interesting question is to know if the same holds for polynomial time on-line algorithms. Theorem 2 induces that any improvement dealing with the approximation of WHG would immediately induce an improvement of the competitive ratio that can be guaranteed in polynomial-time. What about the converse? In [1] a reduction from WIS to $LWIS(2)$ is proposed, with improvement of the ratio allowing to show that any improvement of $LWIS(2)$'s competitive ratio would imply an improvement of WIS's approximation ratio. A consequence is an hardness result for polynomial-time algorithms. A generalization of this result to $LWIS(t)$ or, more generally, the design of such reductions from off-line to on-line seems to be a fruitful research area.

References

1. Demange, M.: Reduction off-line to on-line: an example and its applications. Yugoslav Journal of Operations Research **13**(1), 3-24, 2003
2. Demange, M., Paschos, V. Th.: Improved approximations for weighted and unweighted graph problems. Theory of Computing Systems. To appear
3. Demange, M., Paradon, X., Paschos, V. Th.: On-line maximun-order induced hereditary subgraph problems. International Transaction in Operational Research, **12**(2), 185-201, 2005
4. Halldòrsson M.M.: Approximations of weighted independent set and heditary subset problems. Proc. 5th Ann. Int. Conf. on Computing and Combinatories, Lecture Notes in Computer Science, Springer-Verlag, 261-270, 1999
5. Crescenzi P., Silvestri R., Trevisan L.:, To weight or not to weight: where is the question? In Proc. 4th Israel Symposium on Theory and Computing and Systems, IEEE Computer Society, 68-77, 1996
6. Garey M.R., Johnson. D.S:, Computers and intractability. A guide to the theory of NP-completness. CA.Freeman, San Francisco, 1979
7. Paradon X.:, Algorithmique on-line. Thèse de doctorat, Université Paris Dauphine, 2000 (in french)